CLAIMS

WHAT IS CLAIMED IS:

1. An apparatus for performing computations comprising:

a chaining controller;

a plurality of computational devices;

wherein a first chaining subset of the plurality of computational devices includes at least two of the plurality of computational devices; and

wherein the chaining controller is configured to instruct the first chaining subset to operate as a first computational chain.

2. The apparatus of Claim 1, wherein the plurality of computational devices comprises exponentiators, whereby the first computational chain comprises a first exponentiation chain.

3. The apparatus of Claim 2, further comprising a hardware state controller for each exponentiator of the first exponentiation chain, wherein each hardware state controller includes replicated fanout control logic.

4. The apparatus of Claim 3, wherein the replicated fanout control logic is configured to allow exponentiators of the first exponentiation chain to chain without delay due to high fanout.

5. The apparatus of Claim 3, wherein the replicated fanout control logic is configured such that state machines of the first exponentiation chain sequence efficiently.

6. The apparatus of Claim 2,

wherein each exponentiator further comprises a custom multiplier datapath; and

wherein each custom multiplier datapath is configured so that the length of its longest wire is short.

7. The apparatus of Claim 6, wherein the custom multiplier datapaths of chained exponentiators are physically mirrored to each other so that the wire length between the two is short.

8. The apparatus of Claim 6, wherein the custom multiplier datapath has a serpentine layout so that the wire length between the most separated adjacent data locations is short.

9. The apparatus of Claim 2, wherein the number of exponentiators in the plurality of exponentiators equals 2^k, wherein k is a nonnegative integer.

10. The apparatus of Claim 9, wherein k equals 2.

11. The apparatus of Claim 9, wherein each exponentiator is adapted to exponentiate a 512-bit number.

12. The apparatus of Claim 2, wherein the number of exponentiators in the exponentiation chain equals 2^k, wherein k is a positive integer.

13. The apparatus of Claim 12, further comprising:

a second exponentiation chain;

wherein a second chaining subset of the plurality of exponentiators includes at least two of the plurality of exponentiators;

wherein the chaining controller is configured to instruct the second chaining subset to operate as a second exponentiation chain.

wherein no exponentiator of the first exponentiation chain is part of the second exponentiation chain.

14. The apparatus of Claim 2, wherein each exponentiator further comprises:

a cleave/merge engine;

wherein the cleave/merge engine is configured to:

receive AA, which is a 2w-bit number;

calculate A_1 and A_2, which are two w-bit numbers based on AA; and

output A_1 and A_2;

wherein the cleave/merge engine is also configured to:

receive B_1 and B_2, which are two w-bit numbers;

calculate BB, which is a 2w-bit number based on B_1 and B_2; and

output BB;

wherein exponentiation of AA yields BB;

wherein exponentiation of A_1 yields B_1;

wherein exponentiation of A_2 yields B_2; and

wherein w is a positive integer.

15. The apparatus of Claim 14, wherein A_1 and A_2 are calculated from AA, and BB is calculated from B_1 and B_2, using a scalable Chinese Remainder Theorem implementation.

16. The apparatus of Claim 15,

wherein each exponentiator is adapted to perform 1024-bit exponentiation;

wherein, if 2048-bit exponentiation is required, the chaining controller causes the first exponentiation chain to comprise two exponentiators; and

wherein, if 4096-bit exponentiation is required, the chaining controller causes the first exponentiation chain to comprise four exponentiators.

17. A system for computing comprising:

a computing device;

at least one apparatus of Claim 1; and

wherein the computing device is configured to use the apparatus of Claim 1 to perform computations.

18. A method for performing computations comprising:

loading argument X into session memory;

loading argument K into session memory;

cleaving X mod P to compute X_P;

cleaving X mod Q to compute X_Q;

exponentiating X_P to compute C_P;

exponentiating X_Q to compute C_Q;

merging C_P and C_Q to compute C; and

retrieving C from the session memory.

19. The method of Claim 18, further comprising:

selecting one session controller of 32 available session controllers;

setting the busy bit for the one session controller;

wherein the argument X is a 1024-bit number;

wherein C is a 1024-bit number; and

clearing the busy bit for the one session controller.

20. The method of Claim 18, further comprising:

selecting two session controllers of 32 available session controllers;

setting the busy bits for the two session controllers

wherein loading argument X into session memory includes:

loading part of the argument X into the session memory of one of the two session controllers;

loading the remainder of the argument X into the session memory of the other of the two session controllers;

wherein the argument X is a 2048-bit number;

wherein C is a 2048-bit number; and

clearing the busy bits for the two session controllers.

21. The method of Claim 18, further comprising:

selecting four session controllers of 32 available session controllers;

setting the busy bits for the four session controllers

wherein loading argument X into session memory includes:

loading a first part of the argument X into the session memory of a first of the four session controllers;

loading a second part of the argument X into the session memory of a second of the four session controllers;

loading a third part of the argument X into the session memory of a third of the four session controllers;

loading the remaining of the argument X into the session memory of a fourth of the four session controllers;

wherein the argument X is a 4096-bit number;

wherein C is a 4096-bit number; and

clearing the busy bits for the four session controllers.

22. The method of Claim 18,

wherein the cleaving X mod P comprises:

setting A[513:0] = X[1023:510];

calculating Z[1026:0] = A[513:0] × μP[512:0], wherein μP[512] = 1;

setting B[513:0] = Z[1026:512];

setting C[513:0] = X[513:0];

calculating Y[1025:0] = B[513:0] × P[511:0];

setting D[513:0] = Y[513:0];

calculating E[513:0] = C[513:0] - D[513:0];

if E > P then calculating E = E - P;

if E > P then E = E - P; and

setting X_P = E[511:0] as the result of the cleaving X mod P, whereby X_P equals X mod P; and

wherein the cleaving X mod Q comprises:

setting A[513:0] = X[1023:510];

calculating Z[1026:0] = A[513:0] × μQ[512:0], wherein μQ[512] = 1;

setting B[513:0] = Z[1026:512];

setting C[513:0] = X[513:0];

calculating Y[1025:0] = B[513:0] × Q[511:0];

setting D[513:0] = Y[513:0];

calculating E[513:0] = C[513:0] - D[513:0];

if E > Q then calculating E = E - Q;

if E > Q then E = E - Q; and

setting X_Q = E[511:0] as the result of the cleaving X mod Q, whereby X_Q equals X mod Q.

23. The method of Claim 18, wherein merging C_P and C_Q to compute C comprises:

if $C_P > P$ then calculating $C_P = C_P - P$;

if $C_Q > Q$ then calculating $C_Q = C_Q - Q$;

calculating $A[512:0] = C_Q[511:0] - C_P[511:0]$;

if $A < 0$ then calculating $A[511:0] = A[511:0] + Q[511:0]$;

calculating $B[1023:0] = A[511:0] \times P^{-1}[511:0]$;

calculating $D[511:0] = \text{Cleave } B[1023:0] \bmod Q[511:0]$, wherein $\mu Q[512] = 1$;

calculating $E[1023:0] = D[511:0] \times P[511:0]$;

calculating $C[1023:0] = E[1023:0] + Cp[511:0]$; and

wherein $C[1023:0]$ is the result of merging C_P and C_Q.